Python environment of legged chain for reinforcement learning

The function environment.step in environment.py computes one step of the legged chain with properties as defined in parameters.py and takes inputs of current time, state, action, contact info, and parameters. It returns the final time of the step, the new state, the new contact configuration, the reward, and within-step states. Note that the leg sequences are fixed in the current version and the action merely defines the magnitude of leg torques and body bending torques as a function of time. The question to be answered is therefore: Given the leg sequence as defined in parameters.py, what are the forces that maximize the reward as defined in reward.py?

Run runExample.py for a test case with constant leg torques and no bending torques that leads to nice locomotion. Currently animation is done in matlab, so run animation.m in matlab folder.

environment.py

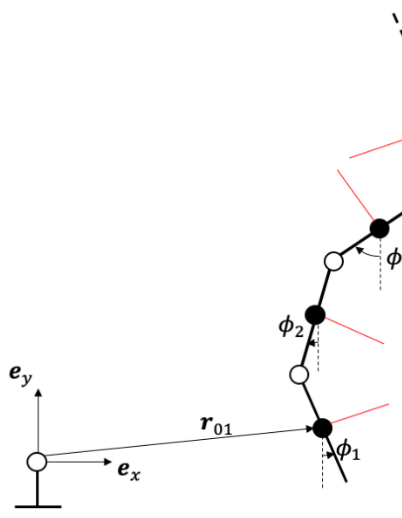
```
def step(time, state, action, cInfo, p):
…

return [tEvent, newState, cInfo, rew, tSteps, sol]
```

Time	Scalar value of current simulation time
State	Array of state which is composed of the generalized position and velocities of the system: $state = [\boldsymbol{q}, \boldsymbol{u}]^T$ which is $\mathbb{R}^{2(n+2)\times 1}$ dimensional (n elements in chain, 2 degrees of freedom for position in the plane)
Action	Array of leg torques and bending torques. Action is $\mathbb{R}^{2n-1\times 1}$ 2n-1 dimensional (n leg torques for each element in body, n-1 bending torques between the elements
Contact info	Array cInfo contains in first row all current closed leg ground contacts, the corresponding leg angles, element center of mass positions, and leg vectors.
Parameters	Class parameters and instance p which contains all system-relevant parameters. See below
Reward	Scalar reward defined in reward.py
Within-step states	tSteps contains time information and sol state information. There are various options for numerical solvers and the fastest (though least accurate) is the explicit Euler solver. Change accuracy with time step p.dt in parameters.py. Scipy integrators can also be uncommented and used if needed.

Problem parameters

Note that code was nondimensionalized with f, m, l, i.e. the frequency of the travelling leg wave, the element mass, and the element length. The problem is then defined by the following parameters

- n number of segments
- α initial leg angle
- β non-dimensional moment of inertia
- γ body length to leg length ratio (l/L)
- ϕ inter segment leg phase difference
- dc leg duty cycle
- $\bar{T}$ non-dimensional leg torque
- $\bar{k}$ non-dimensional inter-segment stiffness
- $\bar{d}$ non-dimensional inter-segment damping



Generalized position $\boldsymbol{q}$ and velocity $\boldsymbol{u}$

$$\boldsymbol{q} = [x, y, \theta_1, \theta_2, \dots, \theta_n]^T$$

$$\boldsymbol{u} = [\dot{x}, \dot{y}, \dot{\theta}_1, \dot{\theta}_2, \dots, \dot{\theta}_n]^T$$

Vector to first element on chain

$$\boldsymbol{r}_1 = \begin{pmatrix} x \\ y \end{pmatrix};\ \dot{\boldsymbol{r}}_1 = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix};\ \ddot{\boldsymbol{r}}_1 = \begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix}$$

Generalized force direction of first element

$$\mathbf{J}_1 = \begin{bmatrix} 1 & 0 & 0 \dots \\ 0 & 1 & 0 \dots \end{bmatrix}$$

Vector to second element on chain (l being element length)

$$\boldsymbol{r}_2 = \boldsymbol{r}_1 + \frac{l_1}{2}\begin{pmatrix} -\sin\theta_1 \\ \cos\theta_1 \end{pmatrix} + \frac{l_2}{2}\begin{pmatrix} -\sin\theta_2 \\ \cos\theta_2 \end{pmatrix};\ \dot{\boldsymbol{r}}_2 = \dot{\boldsymbol{r}}_1 + \frac{l_1}{2}\begin{pmatrix} -\dot{\theta}_1\cos\theta_1 \\ -\dot{\theta}_1\sin\theta_1 \end{pmatrix} + \frac{l_2}{2}\begin{pmatrix} -\dot{\theta}_2\cos\theta_2 \\ -\dot{\theta}_2\sin\theta_2 \end{pmatrix};\ \ddot{\boldsymbol{r}}_2$$

$$= \ddot{\boldsymbol{r}}_1 + \frac{l_1}{2}\begin{pmatrix} -\ddot{\theta}_1\cos\theta_1 + \dot{\theta}_1{}^2\sin\theta_1 \\ -\ddot{\theta}_1\sin\theta_1 - \dot{\theta}_1{}^2\cos\theta_1 \end{pmatrix} + \frac{l_2}{2}\begin{pmatrix} -\ddot{\theta}_2\cos\theta_2 + \dot{\theta}_2{}^2\sin\theta_2 \\ -\ddot{\theta}_2\sin\theta_2 - \dot{\theta}_2{}^2\cos\theta_2 \end{pmatrix}$$

Generalized force direction of second element

$$\mathbf{J}_2 = \mathbf{J}_1 + \begin{bmatrix} \dots 0 & -\frac{l_1}{2}\cos\theta_2 & -\frac{l_2}{2}\cos\theta_2 & 0 \dots \\ \dots 0 & -\frac{l_1}{2}\sin\theta_2 & -\frac{l_2}{2}\sin\theta_2 & 0 \dots \end{bmatrix}$$

Therefore, for the kth element:

$$\boldsymbol{r}_k = \boldsymbol{r}_{k-1} + \frac{l_{k-1}}{2}\begin{pmatrix} -\sin\theta_{k-1} \\ \cos\theta_{k-1} \end{pmatrix} + \frac{l_k}{2}\begin{pmatrix} -\sin\theta_k \\ \cos\theta_k \end{pmatrix}$$

$$\dot{\boldsymbol{r}}_k = \dot{\boldsymbol{r}}_{k-1} + \frac{l_{k-1}\dot{\theta}_{k-1}}{2}\begin{pmatrix} -\cos\theta_{k-1} \\ -\sin\theta_{k-1} \end{pmatrix} + \frac{l_k\dot{\theta}_k}{2}\begin{pmatrix} -\cos\theta_k \\ -\sin\theta_k \end{pmatrix}$$

$$\ddot{\boldsymbol{r}}_k = \ddot{\boldsymbol{r}}_{k-1} + \frac{l_{k-1}\ddot{\theta}_{k-1}}{2}\begin{pmatrix} -\cos\theta_{k-1} \\ -\sin\theta_{k-1} \end{pmatrix} + \frac{l_k\ddot{\theta}_k}{2}\begin{pmatrix} -\cos\theta_k \\ -\sin\theta_k \end{pmatrix} + \frac{l_{k-1}\dot{\theta}_{k-1}^2}{2}\begin{pmatrix} \sin\theta_{k-1} \\ -\cos\theta_{k-1} \end{pmatrix} + \frac{l_k\dot{\theta}_k^2}{2}\begin{pmatrix} \sin\theta_k \\ -\cos\theta_k \end{pmatrix}$$

$$\mathbf{J}_k = \begin{bmatrix} \dots 0 & -\frac{l_{k-1}}{2}\cos\theta_{k-1} & -\frac{l_k}{2}\cos\theta_k & 0\ \dots \\ \dots 0 & -\frac{l_{k-1}}{2}\sin\theta_{k-1} & -\frac{l_k}{2}\sin\theta_k & 0\ \dots \end{bmatrix} + \mathbf{J}_{k-1}$$

Using the projected Newton-Euler equations the inertial part of the equations of motion follow (assuming $l_k = l, m_k = m\ \forall k$)

$$\sum_{k=1}^{n} \mathbf{J}_k^T\dot{\boldsymbol{p}}_k + \mathbf{J}_{R,k}^T\dot{\boldsymbol{N}}_k = \sum_{k=1}^{n} \mathbf{J}_k^T\ddot{\boldsymbol{r}}_k m_k + \mathbf{J}_{R,k}^T\frac{d}{dt}(\mathbf{J}_{R,k}\dot{\boldsymbol{q}})I_k$$

$$= m(\mathbf{J}_1^T\ddot{\boldsymbol{r}}_1 + \mathbf{J}_2^T\ddot{\boldsymbol{r}}_2 + \cdots) + I\left(\mathbf{J}_{R,1}^T\frac{d}{dt}(\mathbf{J}_{R,1}\dot{\boldsymbol{q}}) + \mathbf{J}_{R,2}^T\frac{d}{dt}(\mathbf{J}_{R,2}\dot{\boldsymbol{q}}) + \cdots\right)$$

$$= m(\mathbf{J}_1^T\mathbf{J}_1 + \mathbf{J}_2^T\mathbf{J}_2 + \cdots)\ddot{\boldsymbol{q}} + m(\mathbf{J}_1^T\dot{\mathbf{J}}_1 + \mathbf{J}_2^T\dot{\mathbf{J}}_2 + \cdots)\dot{\boldsymbol{q}} + I(\mathbf{J}_{R,1}^T\mathbf{J}_{R,1} + \mathbf{J}_{R,2}^T\mathbf{J}_{R,2} + \cdots)\ddot{\boldsymbol{q}}$$

Rewriting for optimized numerical speed (using A matrices, where $\mathbf{A}_l$, $\mathbf{A}_R$ don't change during run time. Symbol $\circ$ for element-wise multiplication)

$$\mathbf{M} = m\mathbf{A}_l \circ \mathbf{A}_{trig} + I\mathbf{A}_R$$

$$\boldsymbol{h} = \left(m\mathbf{A}_l \circ \mathbf{A}_{dtrig}\right)\dot{\boldsymbol{q}}$$

$$\mathbf{A} = \begin{bmatrix} n & n-1 & n-2 \\ n-1 & n-1 & \dots \\ n-2 & n-2 & \dots \end{bmatrix}$$

With $\mathbf{M}$ mass matrix and $\boldsymbol{h}$ vector of gyroscopic accelerations

External forces

Torsion spring between neighboring elements

$$S_1 = -k(\theta_1 - \theta_2)$$

$$S_k = -k(\theta_k - \theta_{k-1}) - k(\theta_k - \theta_{k+1})$$

$$S_n = -k(\theta_n - \theta_{n-1})$$

$$\widehat{\boldsymbol{S}} = \begin{bmatrix} -k & k & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & k & -2k & k & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & k & -k \end{bmatrix}$$

$$\boldsymbol{S} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \widehat{\boldsymbol{S}} \end{bmatrix} \rightarrow \mathbf{0} \ \ is\ 2x2\ matrix$$

Rotational damper between neighboring elements

$$D_1 = -d\left(\dot{\theta}_1 - \dot{\theta}_2\right)$$

$$D_k = -d\left(\dot{\theta}_k - \dot{\theta}_{k-1}\right) - d\left(\dot{\theta}_k - \dot{\theta}_{k+1}\right)$$

$$D_n = -d\left(\dot{\theta}_k - \dot{\theta}_{k-1}\right)$$

$$\widehat{\boldsymbol{D}} = \begin{bmatrix} -d & d & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & d & -2d & d & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & d & -d \end{bmatrix}$$

$$\boldsymbol{D} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \widehat{\boldsymbol{D}} \end{bmatrix} \rightarrow \mathbf{0} \ \ is\ 2x2\ matrix$$

Leg and bending torques

There are two forms of actuation: leg forces and bending. A leg force in element i will cause the centre of mass of the element to accelerate in a circular path around the leg ground contact point. The cause of this force is effectively a torque between leg and body, which must be accounted for. In addition, bending torques can be active which act between neighboring elements. The total generalized force acting due to muscular activity is thus

$$\boldsymbol{F} = \boldsymbol{F}_{leg} + \boldsymbol{F}_{bend}$$

Equations of motion

$$\ddot{\boldsymbol{q}} = \mathbf{M}^{-1}(\boldsymbol{Sq} + \boldsymbol{Du} - \boldsymbol{h} + \boldsymbol{F} + \mathbf{J}_c\boldsymbol{\lambda})$$

With Lagrange multiplier $\boldsymbol{\lambda}$ and generalized constraint force direction $\mathbf{J}_c$ as described below

Constraints (leg ground contacts)

Say constraints (leg ground contact) are active at segments $\boldsymbol{c}_n$ (integers of segment number) on contact points $\boldsymbol{r}_c$ (vector consisting of all current contact point absolute position vectors stacked).

The condition for no-slip contact at point c of a leg at segment n with the ground is

$$\dot{\boldsymbol{r}}_{cn}^T\boldsymbol{r}_{cn} = 0$$

$$(\dot{\boldsymbol{r}}_n - \dot{\boldsymbol{r}}_c)^T\boldsymbol{r}_{nc} = 0$$

$$(\mathbf{J}_n\boldsymbol{u})^T\boldsymbol{r}_{cn} = 0$$

$$\boldsymbol{r}_{cn}^T\mathbf{J}_n\boldsymbol{u} = 0$$

Note that $\boldsymbol{r}_n$ is a vector containing vectors of all center of mass positions stacked and $\mathbf{J}_n$ accordingly. The contact Jacobian is thus

$$\mathbf{J}_c = \boldsymbol{r}_{cn}^T\mathbf{J}_n.$$

The second derivative wrt time is

$$\dot{\boldsymbol{r}}_{cn}^T\mathbf{J}_n\boldsymbol{u} + \boldsymbol{r}_{cn}^T\dot{\mathbf{J}}_n\boldsymbol{u} + \boldsymbol{r}_{cn}^T\mathbf{J}_n\dot{\boldsymbol{u}} = 0$$

$$\mathbf{J}_c\dot{\boldsymbol{u}} + \boldsymbol{u}^T{\mathbf{J}_n}^T\mathbf{J}_n\boldsymbol{u} + \boldsymbol{r}_{cn}^T\dot{\mathbf{J}}_n\boldsymbol{u} = 0$$

Therefore

$$\boldsymbol{\xi} = \boldsymbol{u}^T{\mathbf{J}_n}^T\mathbf{J}_n\boldsymbol{u} + \boldsymbol{r}_{cn}^T\dot{\mathbf{J}}_n\boldsymbol{u}.$$

Defining $\boldsymbol{\Gamma} = \mathbf{J}^T\boldsymbol{F} + \mathbf{J}_R^T\boldsymbol{T} + \ \boldsymbol{Sq} + \boldsymbol{D}\dot{\boldsymbol{q}} - \boldsymbol{h}$ we have that

$$\mathbf{J}_c\dot{\boldsymbol{u}} + \boldsymbol{\xi} = \mathbf{J}_c\mathbf{M}^{-1}(\boldsymbol{\Gamma} + \mathbf{J}_c^T\boldsymbol{\lambda}) + \boldsymbol{\xi} = 0$$

therefore

$$\boldsymbol{\lambda} = -(\mathbf{J}_c\mathbf{M}^{-1}\mathbf{J}_c^T)^{-1}(\mathbf{J}_c\mathbf{M}^{-1}\boldsymbol{\Gamma} + \boldsymbol{\xi})$$

$$\dot{\boldsymbol{u}} = \mathbf{M}^{-1}(\boldsymbol{\Gamma} + \mathbf{J}_c^T\boldsymbol{\lambda}).$$

For non-smooth events we have the impact equations

$$\boldsymbol{u}^+ = \mathbf{M}^{-1}(\boldsymbol{\Gamma} + \mathbf{J}_c^T\boldsymbol{\lambda})dt + \boldsymbol{u}^-$$

$$\mathbf{J}_c\boldsymbol{u}^+ = \mathbf{J}_c\mathbf{M}^{-1}(\boldsymbol{\Gamma} + \mathbf{J}_c^T\boldsymbol{\lambda})dt + \mathbf{J}_c\boldsymbol{u}^- = 0$$

$$d\boldsymbol{\lambda} = -(\mathbf{J}_c\mathbf{M}^{-1}\mathbf{J}_c^T)^{-1}\mathbf{J}_c\boldsymbol{u}^-$$

$$\boldsymbol{u}^+ = \mathbf{M}^{-1}\mathbf{J}_c^T d\boldsymbol{\lambda} + \boldsymbol{u}^-$$